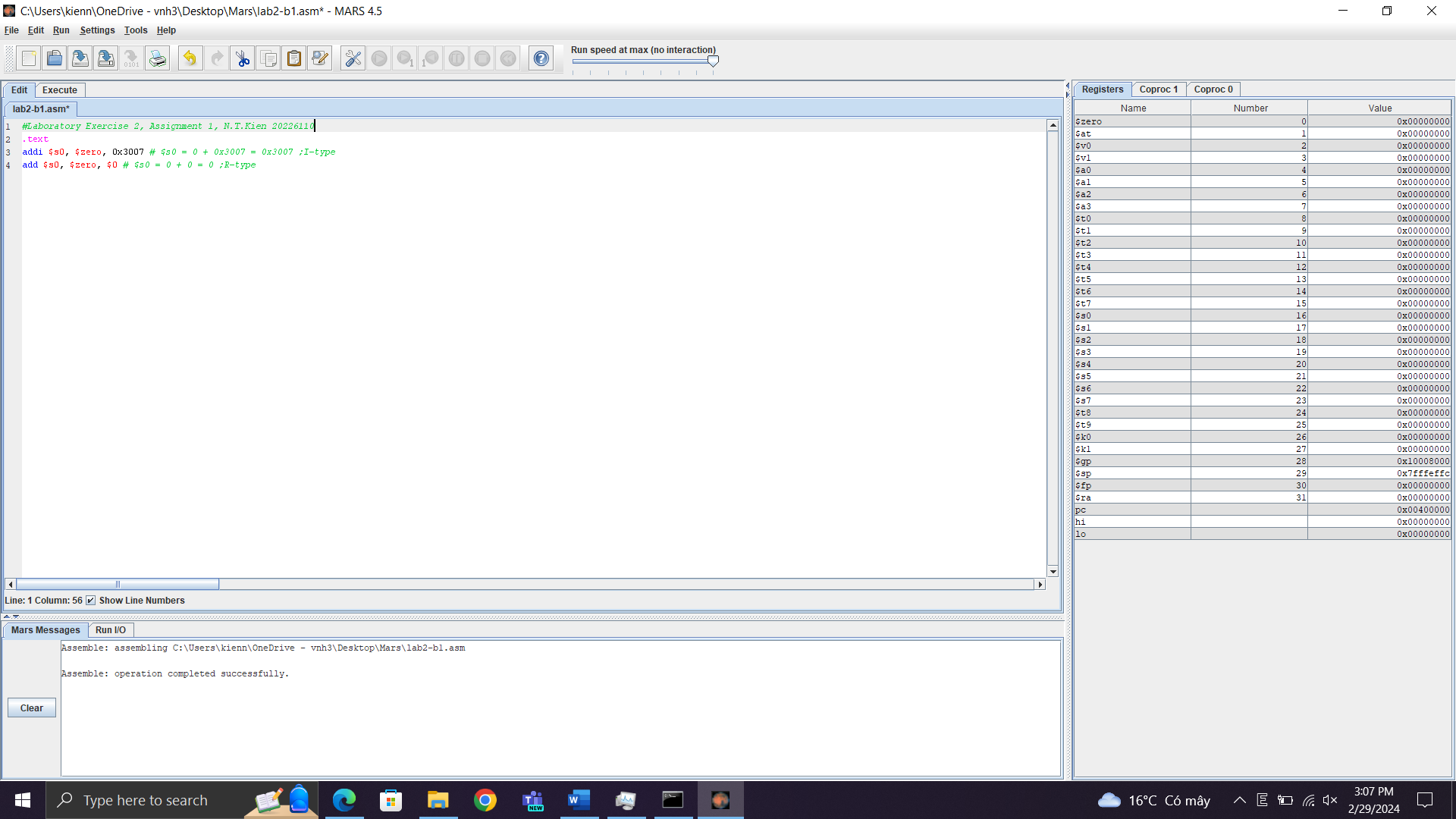
**Báo cáo Thực hành Kiến trúc máy tính Tuần 2**

**Họ và tên: Nguyễn Trung Kiên**

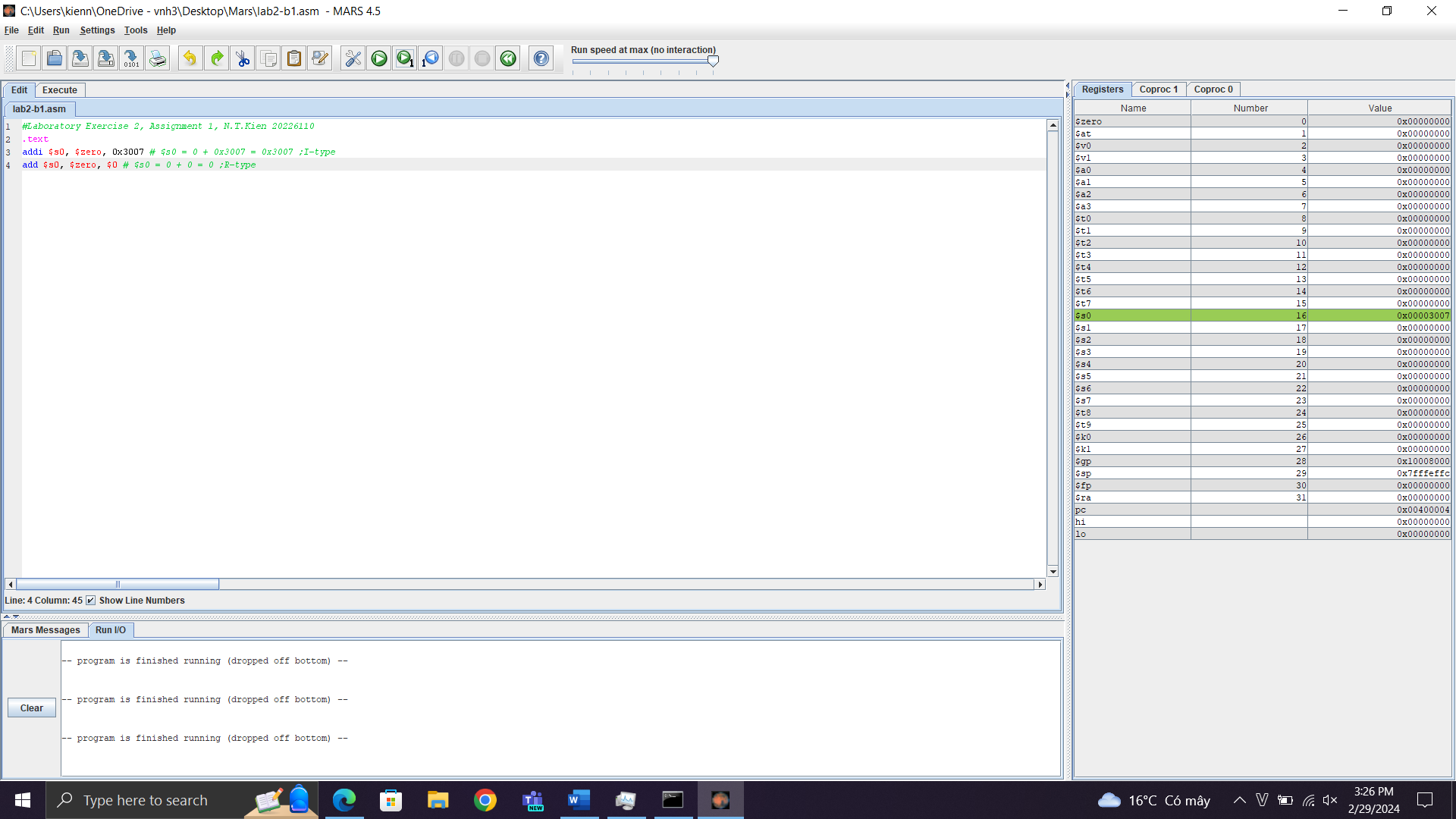
**Lớp: 147794**

**MSSV: 20226110**

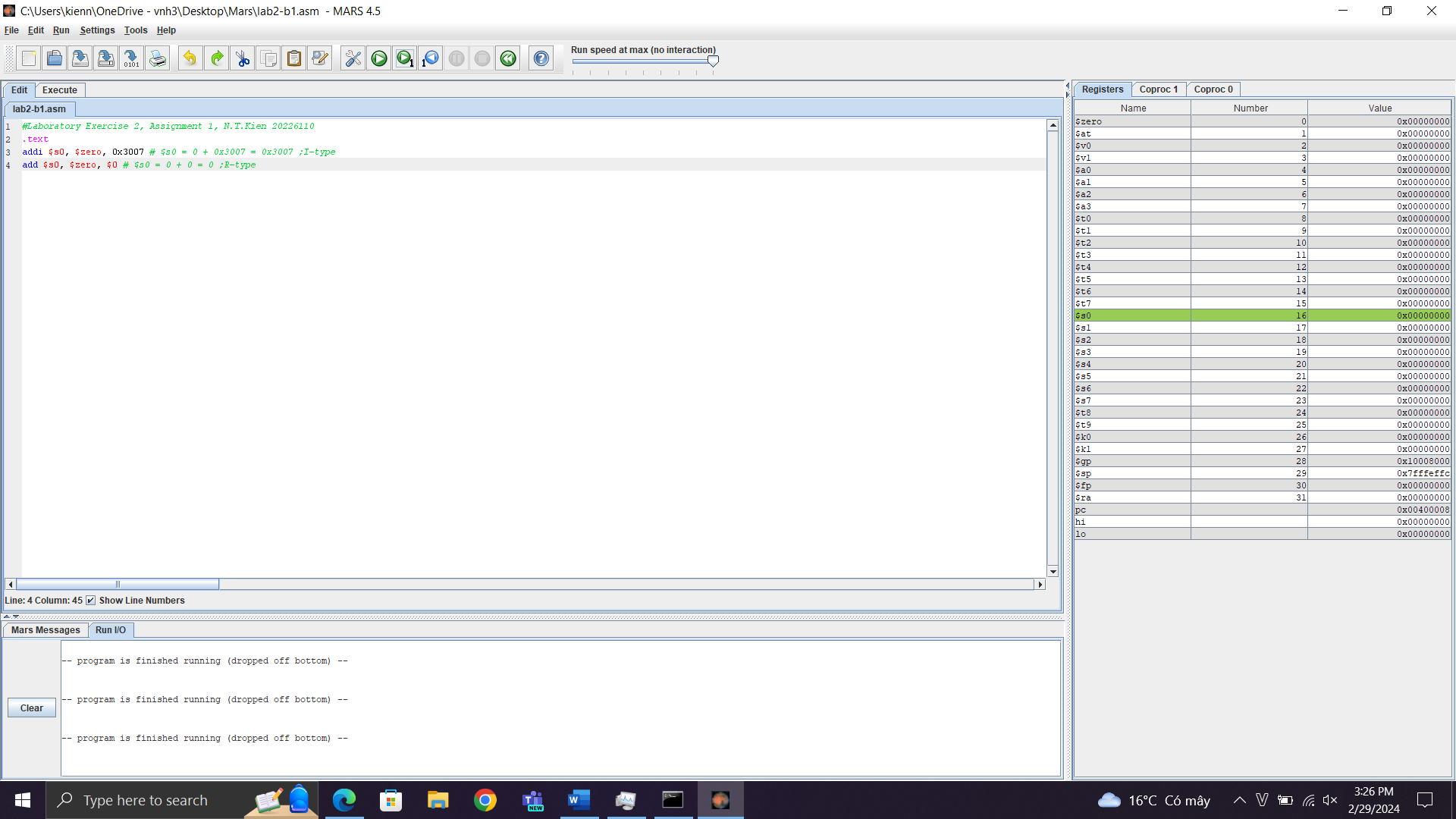
Assignment 1



-Sau lệnh thứ nhất, thanh ghi $0 thay đổi từ 0x00000000 thành 0x00003007.



-Sau lệnh thứ hai, thanh ghi $0 thay đổi từ 0x00003007 thành 0x00000000.



-Thanh ghi pc sẽ tăng thêm một khoảng có giá trị là 0x00000004 sau mỗi lệnh.

So sánh mã máy:

+Lệnh thứ nhất là lệnh I, opcode: 8 => 001000, rs: 0 =>00000, rt: 16=>10000, imm:0x3007=0011 0000 0000 0111

Vậy lệnh máy là: 0010 0000 0001 0000 0011 0000 0000 0111

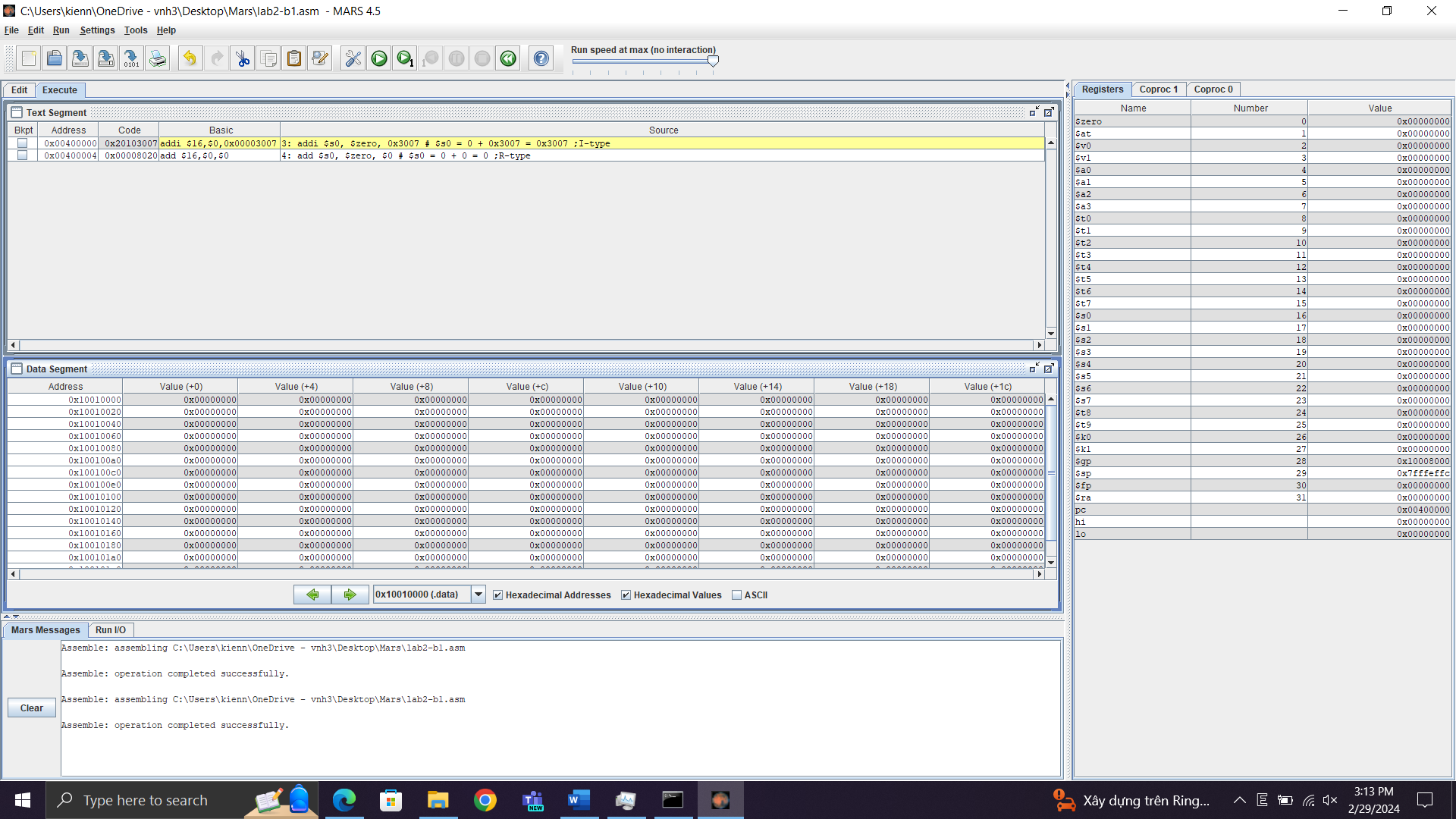
+Lệnh thứ hai là lệnh R, opcode: 0 => 000000, rs: 0 =>00000, rt: 0=>00000, rd: 16=>10000, sh: 0=>00000, fn: 32=>100000

Vậy lệnh máy là: 0000 0000 0000 0000 1000 0000 0010 0000

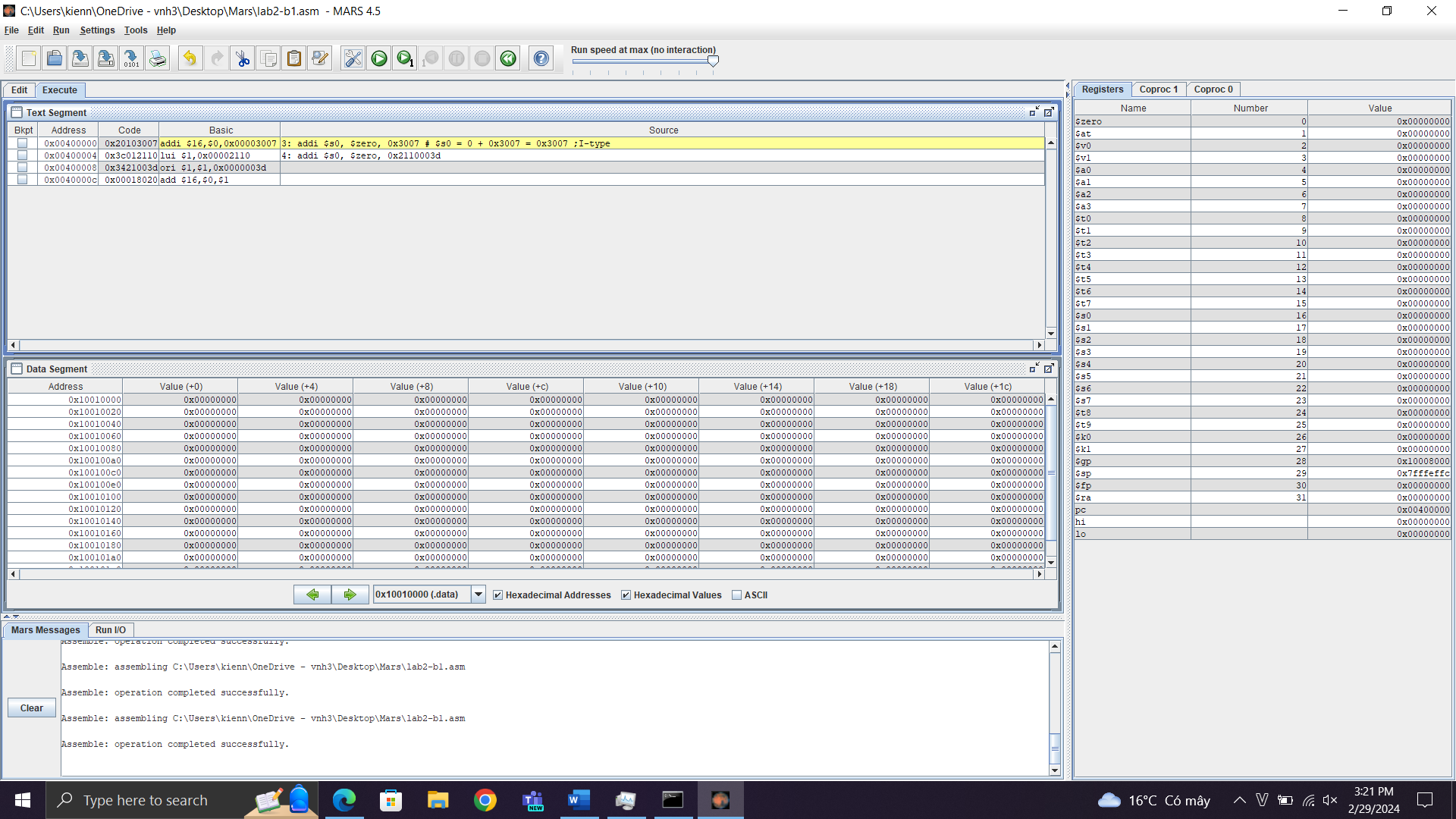
Kết quả này giống như trên ứng dụng MARS.

+ Lệnh thứ nhất: Code 0x20103007, tương đương lệnh máy là 0010 0000 0001 0000 0011 0000 0000 0111.

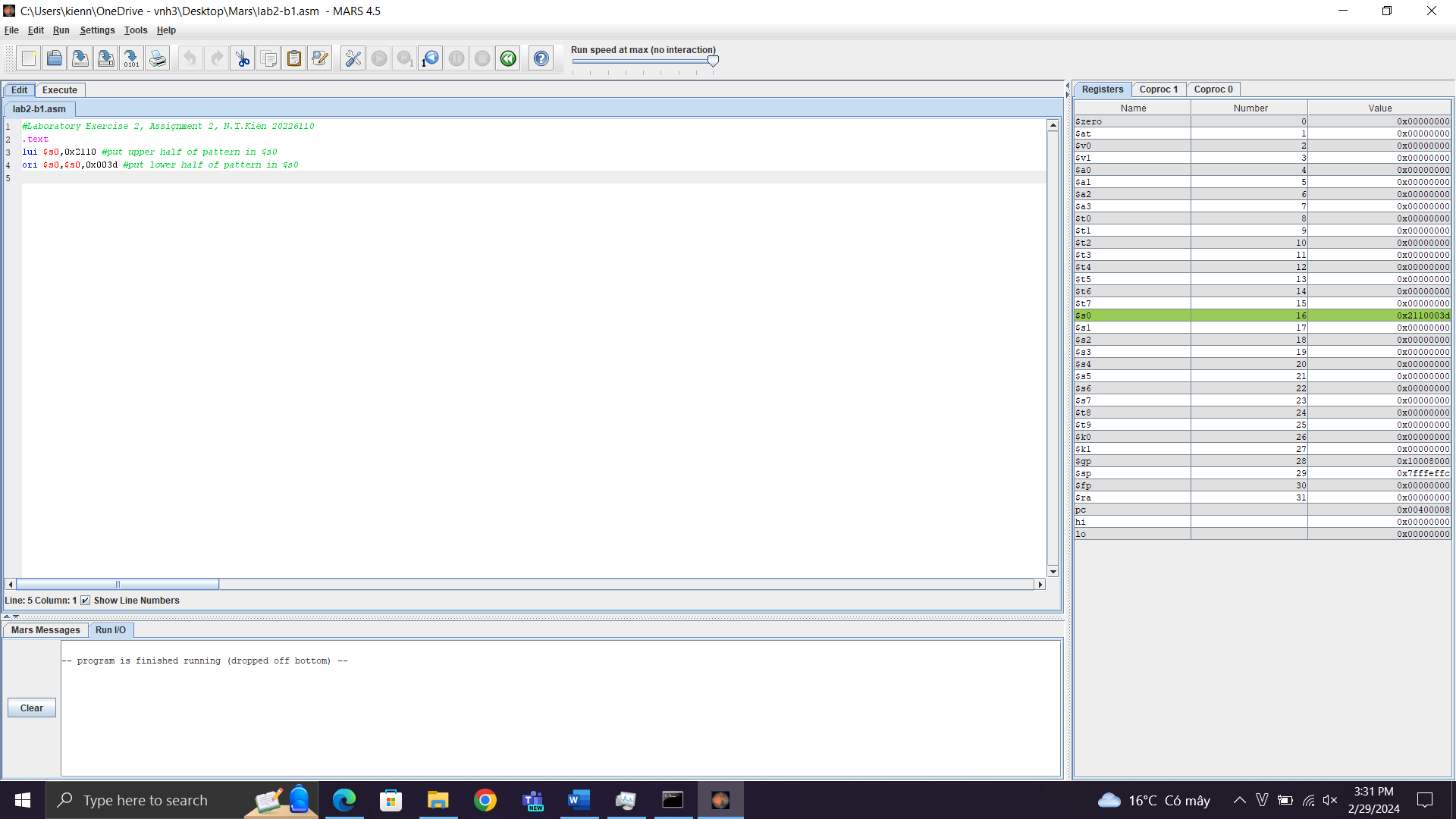
+ Lệnh thứ hai: Code 0x00008020, tương đương lệnh máy là 0000 0000 0000 0000 1000 0000 0010 0000.



+ Nếu sửa lệnh thứ 2 thành addi $s0, $zero, 0x2110003d thì nó thành lệnh gán 32 bit, vì số 0x2110003d lưu trữ ở dạng 32 bit.

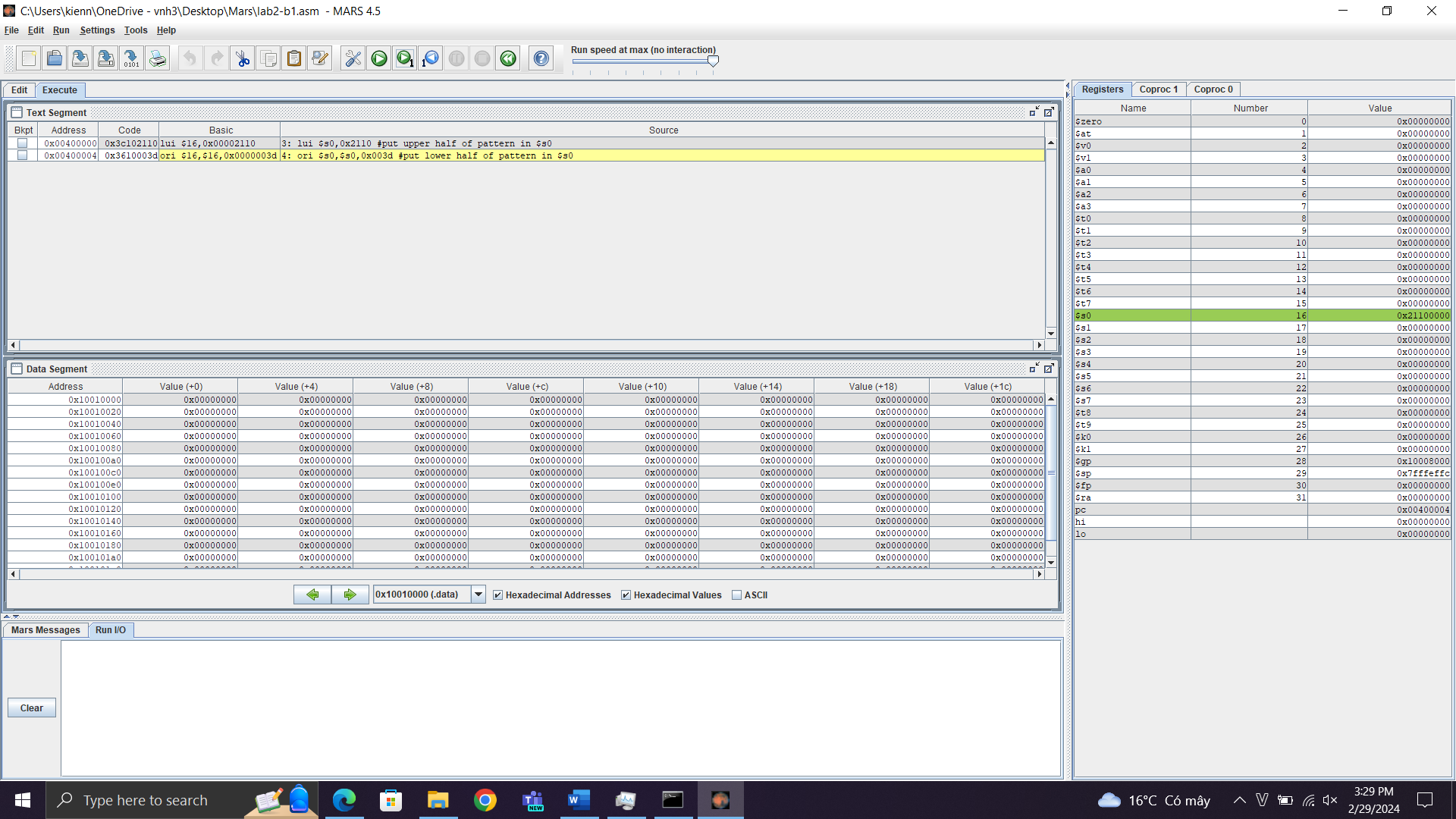


Assignment 2

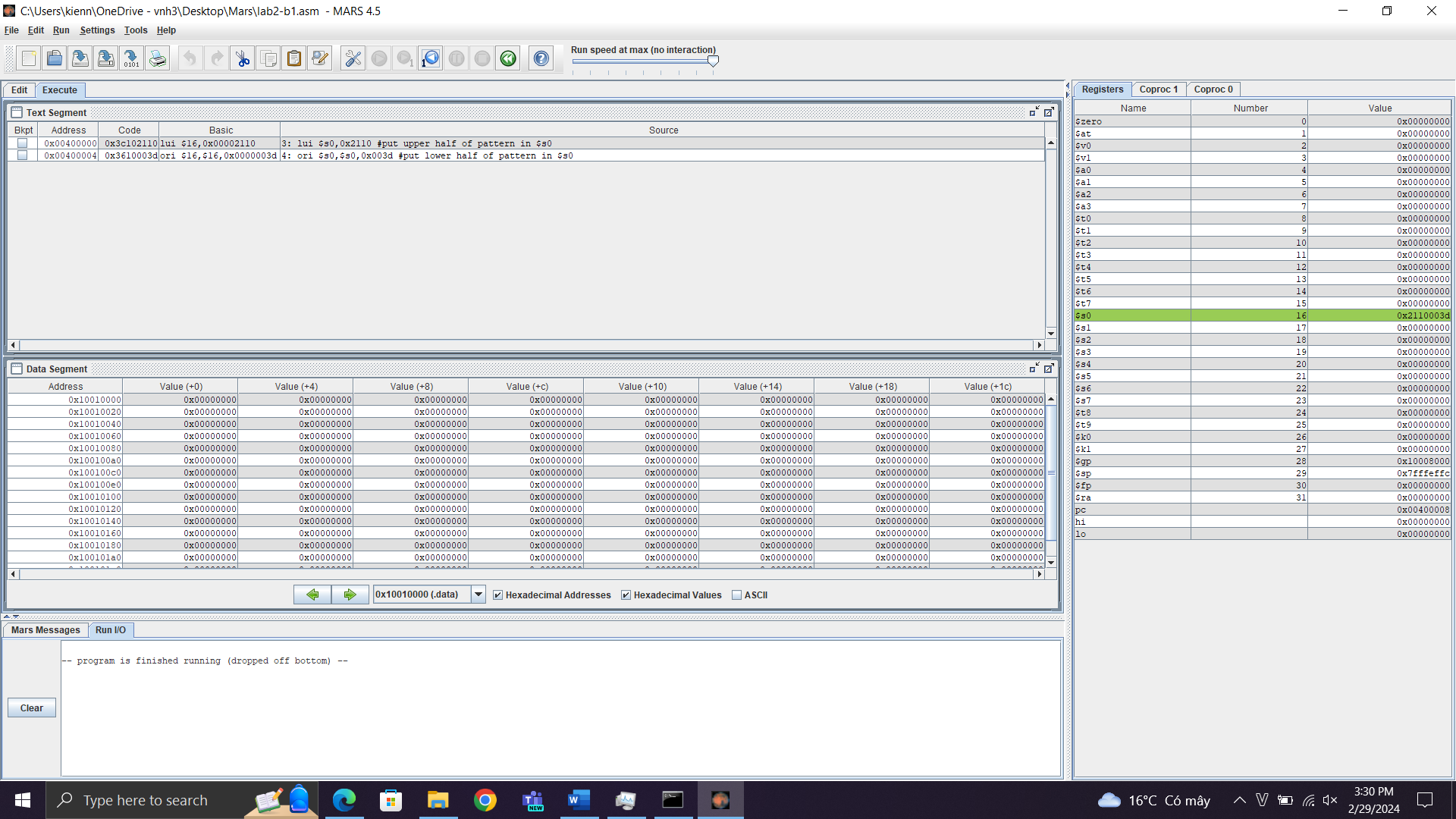


Thanh ghi $s0:

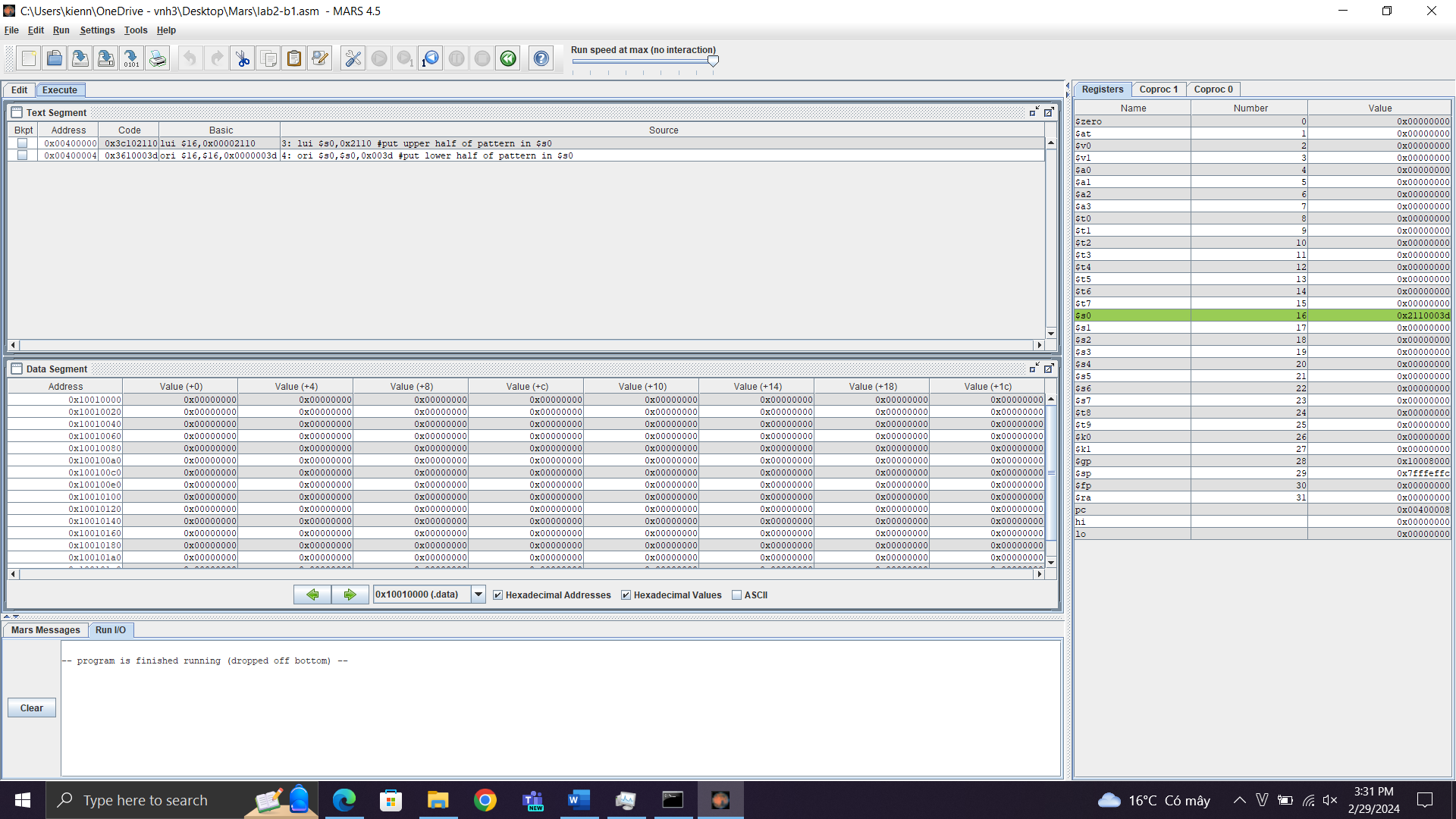
-Sau lệnh thứ nhất thay đổi từ 0x00000000 thành 0x21100000



-Sau lệnh thứ hai thay đổi từ 0x21100000 thành 0x2110003d

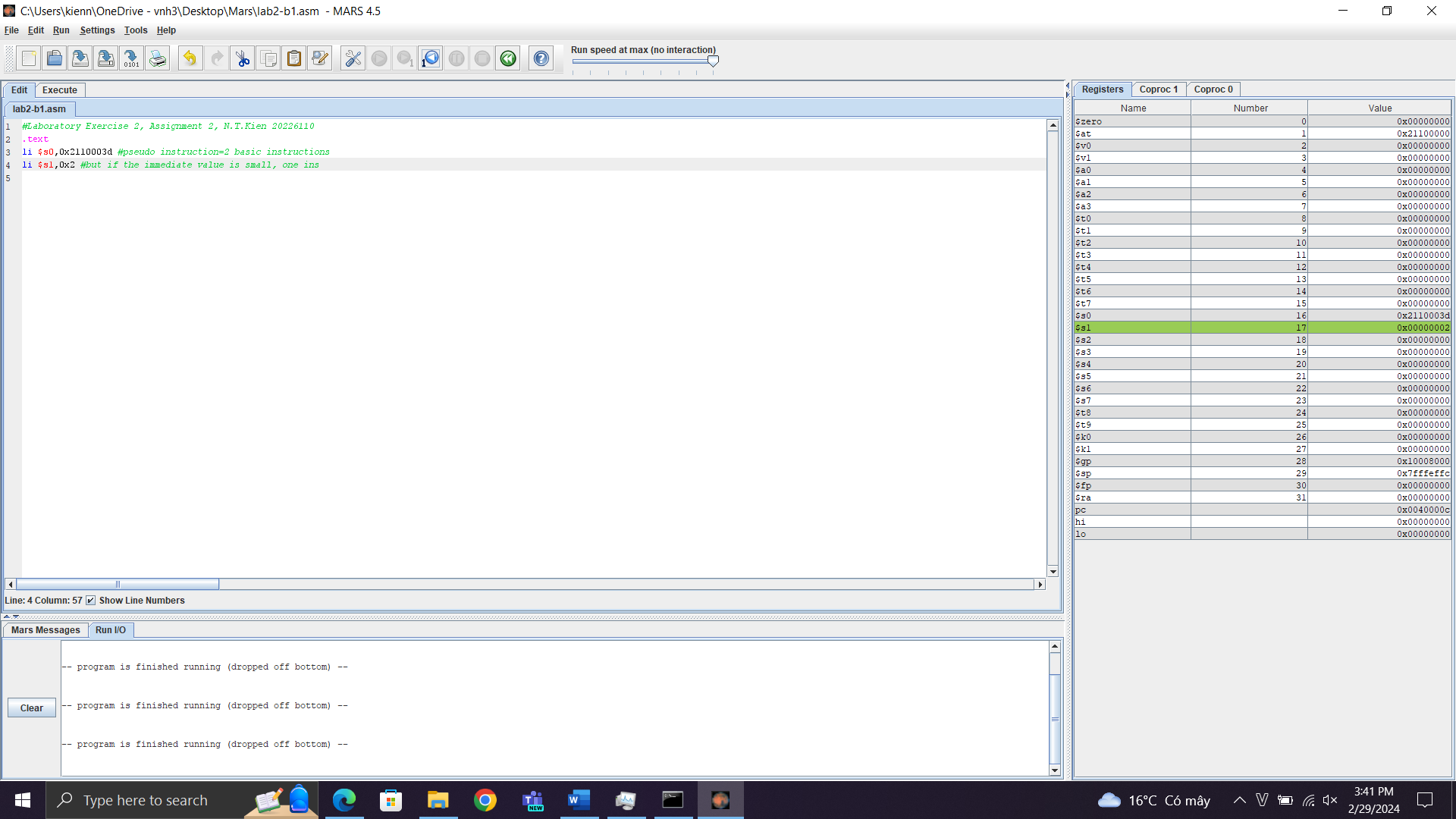


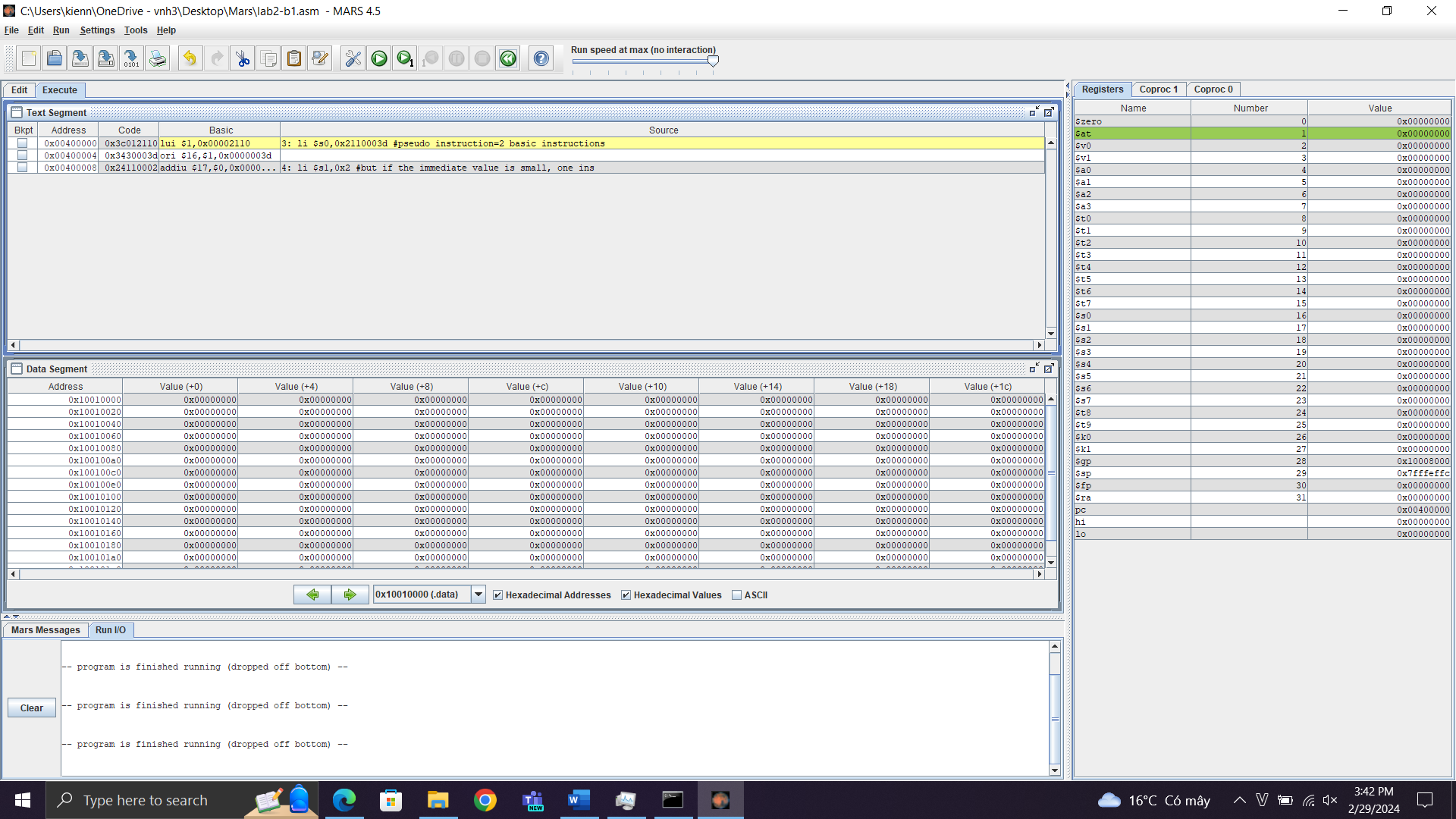
Thanh ghi pc tăng thêm một khoảng có giá trị là 0x00000004 sau mỗi lệnh



* Các byte đầu tiên ở vùng lệnh trùng với cột Code trong cửa sổ Text Segment ở phần thực thi.

Assignment 3





Giải thích điều bất thường:

+ Lệnh li thứ nhất tách ra thành 2 lệnh là lệnh lui và lệnh ori.

+ Lệnh li thứ hai bị biến thành lệnh addiu.

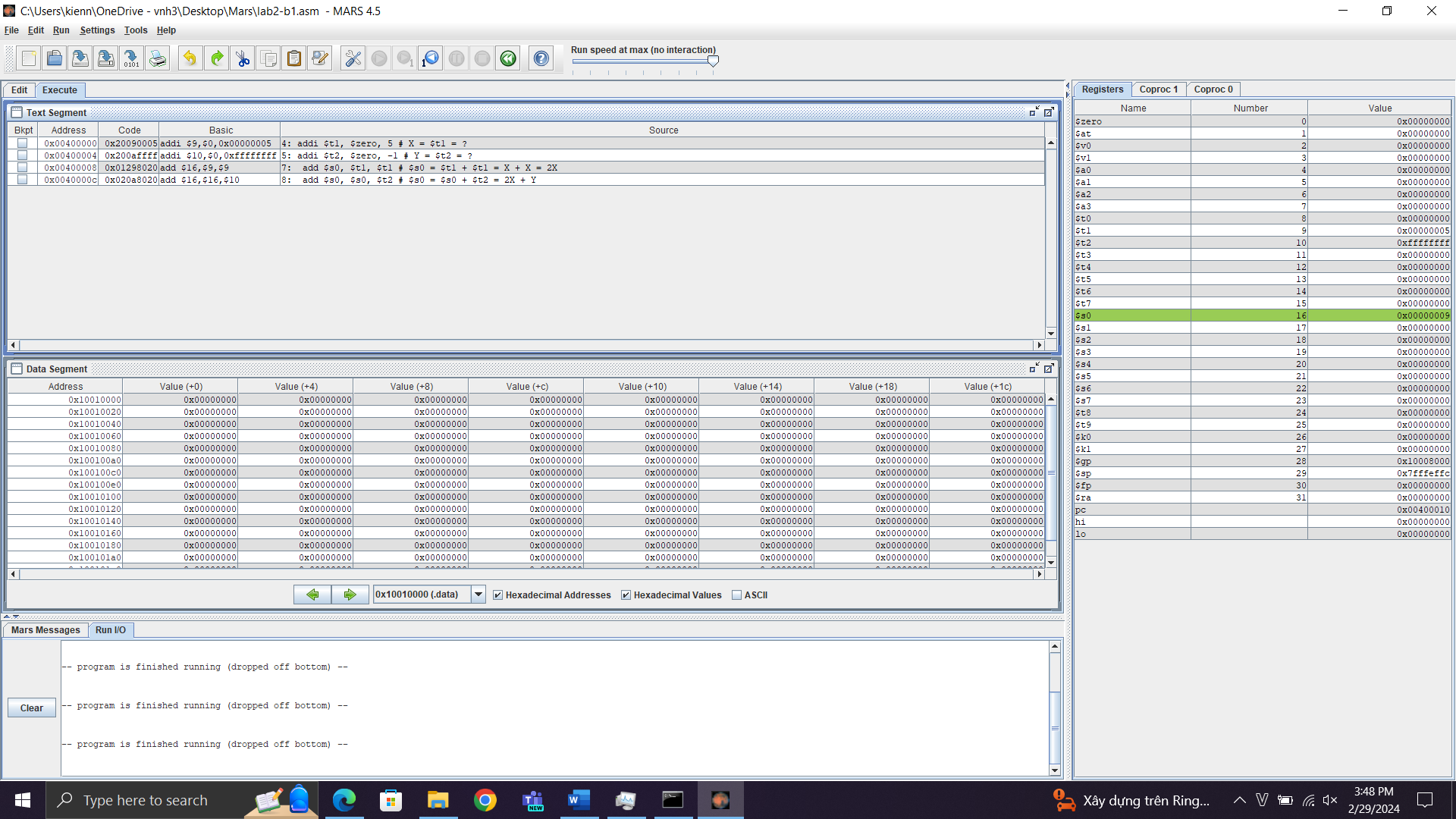
Giải thích: Lệnh li gán giá trị số nguyên bất kỳ, trong trường hợp trên:

+ Lệnh li thứ nhất lệnh li thực hiện gán số 0x2110003d là số loại 32 bit nên tự động tách thành 2 lệnh.

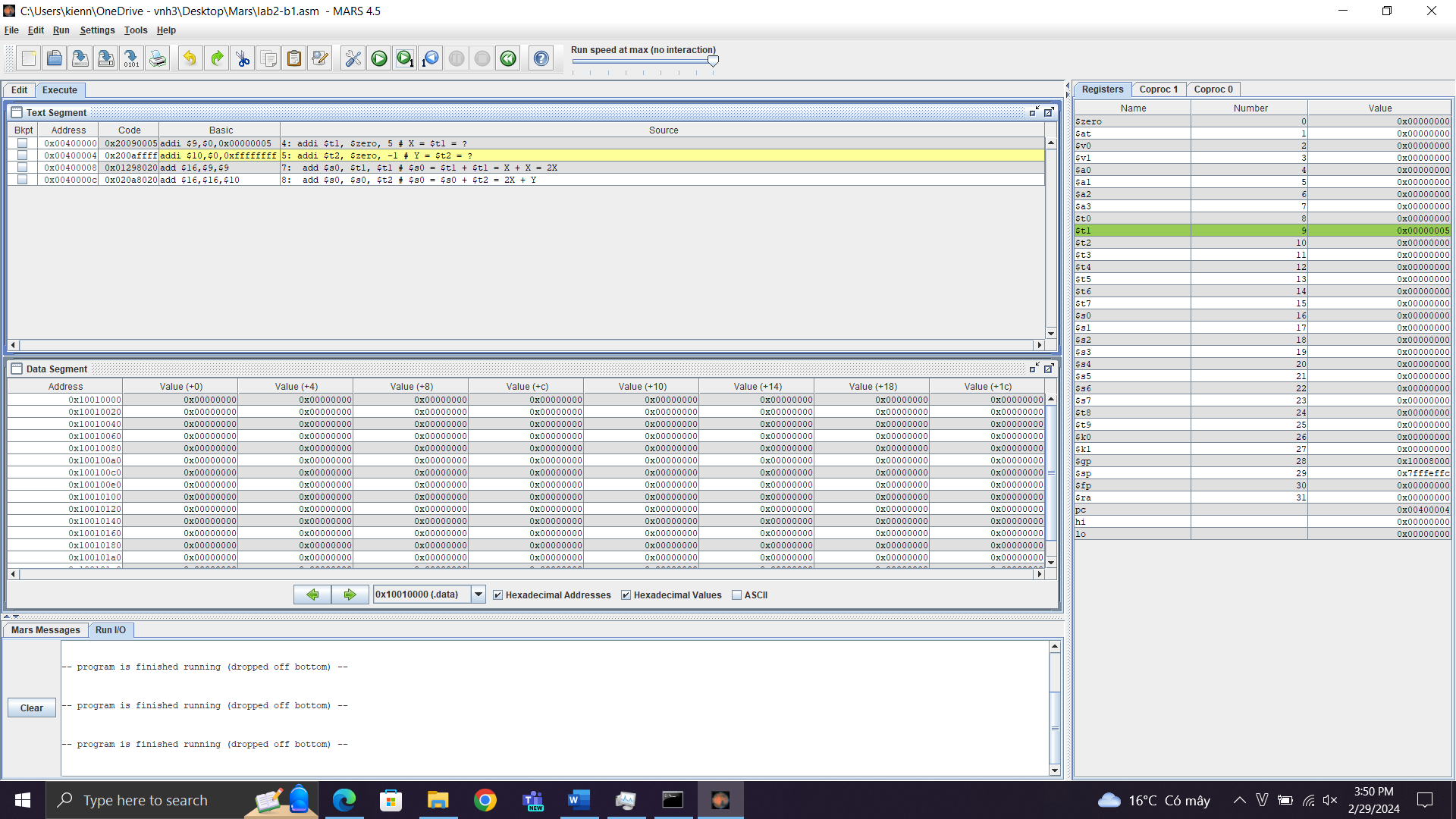
+ Lệnh li thứ hai thực hiện gán số 0x2 là loại số 16 bit không dấu nên chuyển thành lệnh addiu

Assignment 4

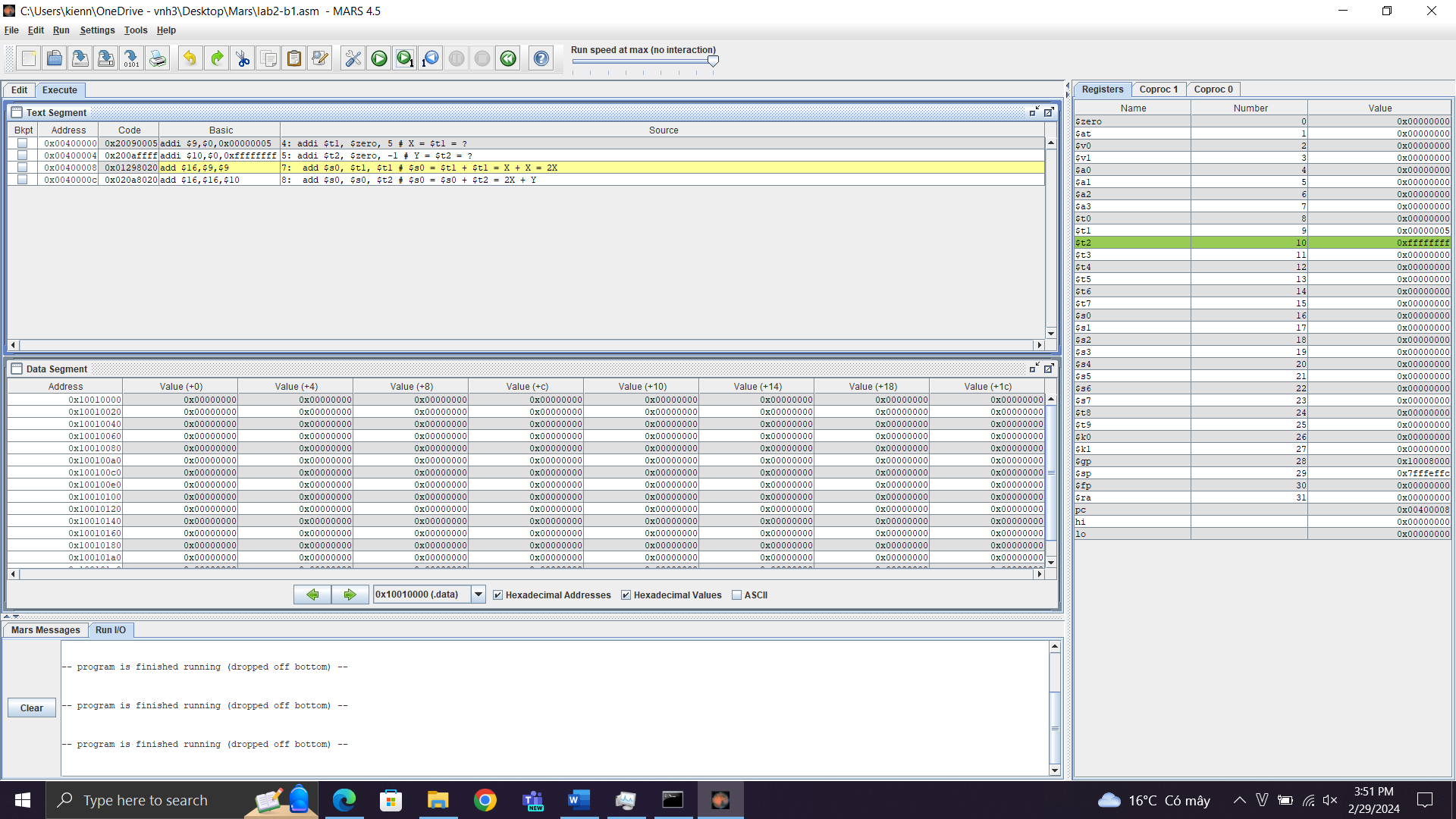




Sau lệnh thứ nhất thanh ghi $t1 thay đổi từ 0x00000000 thành 0x00000005



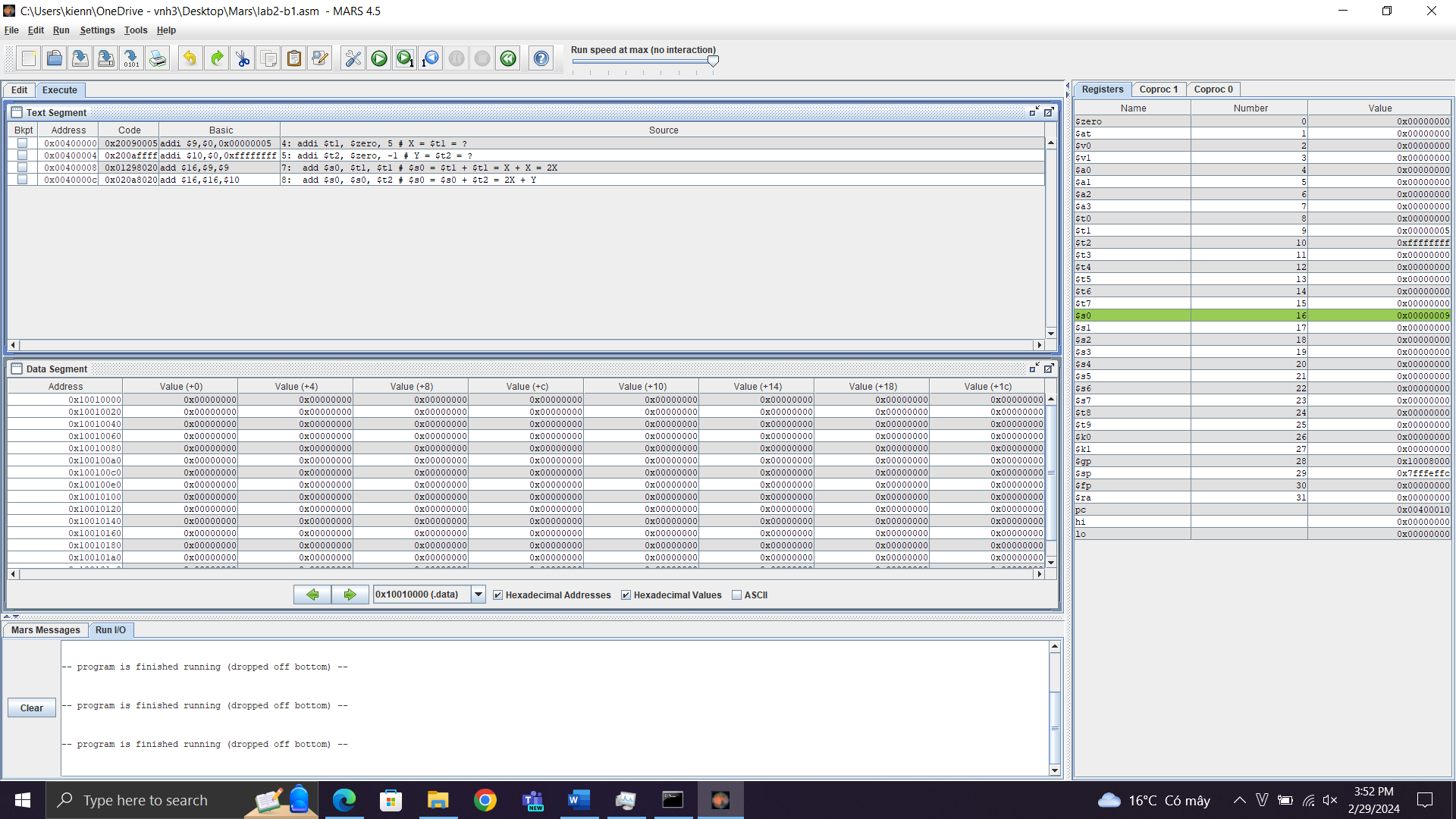
Sau lệnh thứ hai thanh ghi $t2 thay đổi từ 0x00000000 thành 0xffffffff



Sau lệnh thứ ba thanh ghi $s0 thay đổi từ 0x00000000 thành 0x0000000a

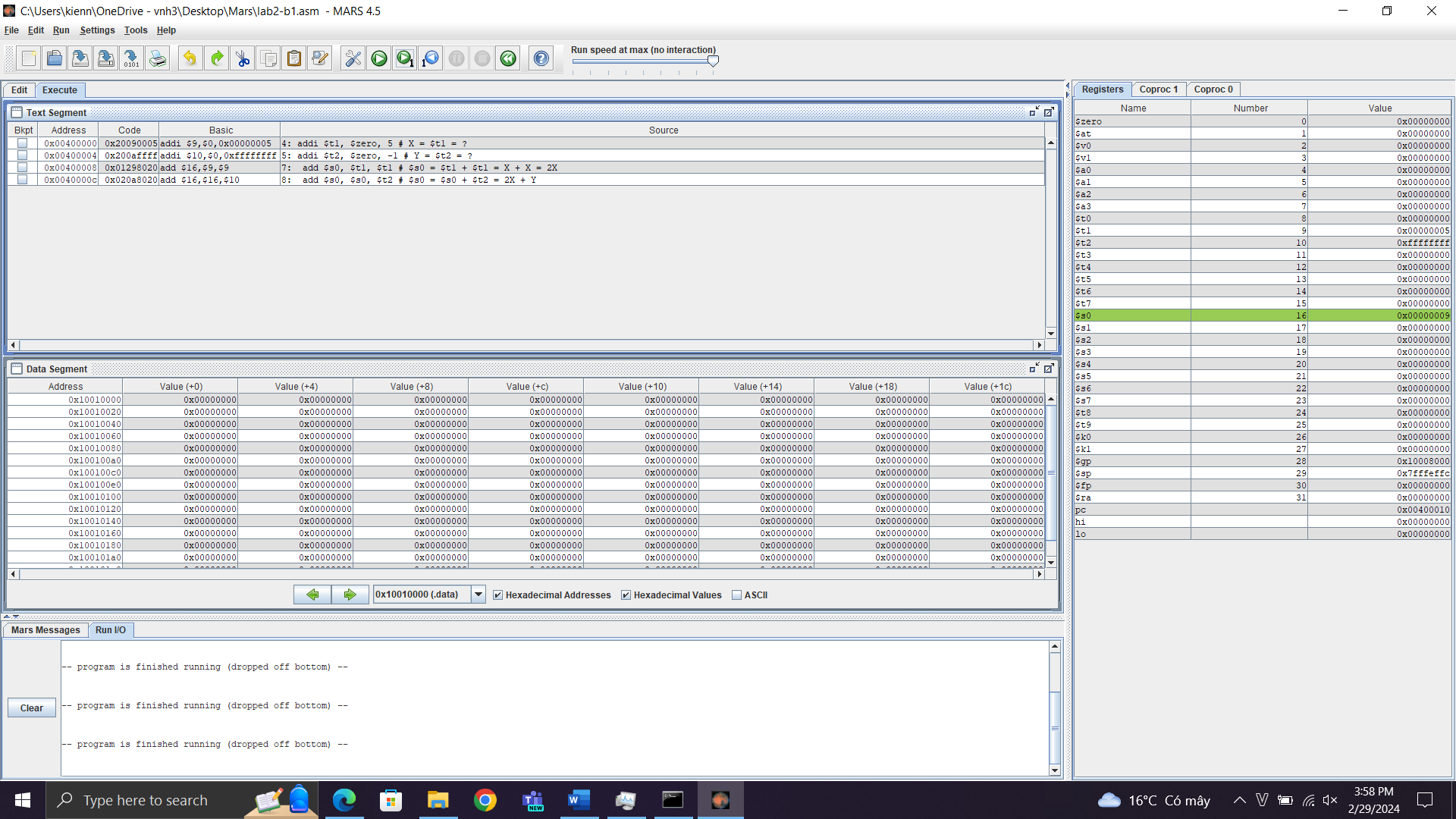


Sau lệnh thứ tư thanh ghi $s0 thay đổi từ 0x0000000a thành 0x00000009



Thanh ghi pc tăng thêm một khoảng có giá trị là 0x00000004 sau mỗi lệnh

* Ta có lệnh addi, sau khi chuyển các yếu tố trong lệnh thành giá trị nhị phân, và ghép các giá trị nhị phân vào khuôn dạng lệnh: opcode-rs-rt-imm thì kết quả ra lệnh mã máy là số nhị phân 32 bit.
* Khi chuyển lệnh mã máy đó sang Hexa thì kết quả giống hệt như trong cột Code và dòng lệnh tương ứng như trên. Do đó nó là điểm tương đồng giữa hợp ngữ và mã máy.



+ Lệnh thứ nhất: addi $t1, $zero, 5 => addi $9, $0, 0x5

opcode: 8 => 001000, rs: $zero => 00000, rt: $t1 => 01001, imm: 0x5 => 0000 0000 0000 0101

=> 0010 0000 0000 1001 0000 0000 0000 0101 => 0x20090005

+Lệnh thứ hai: addi $t2, $zero, -1 => addi $10, $0, 0xffffffff

opcode: 8 => 001000, rs: $0 => 0000, rt: $10 => 01010, imm 0xffffffff => 1111 1111 1111 1111

=> 0010 0000 0000 1010 1111 1111 1111 1111 => 0x200affff

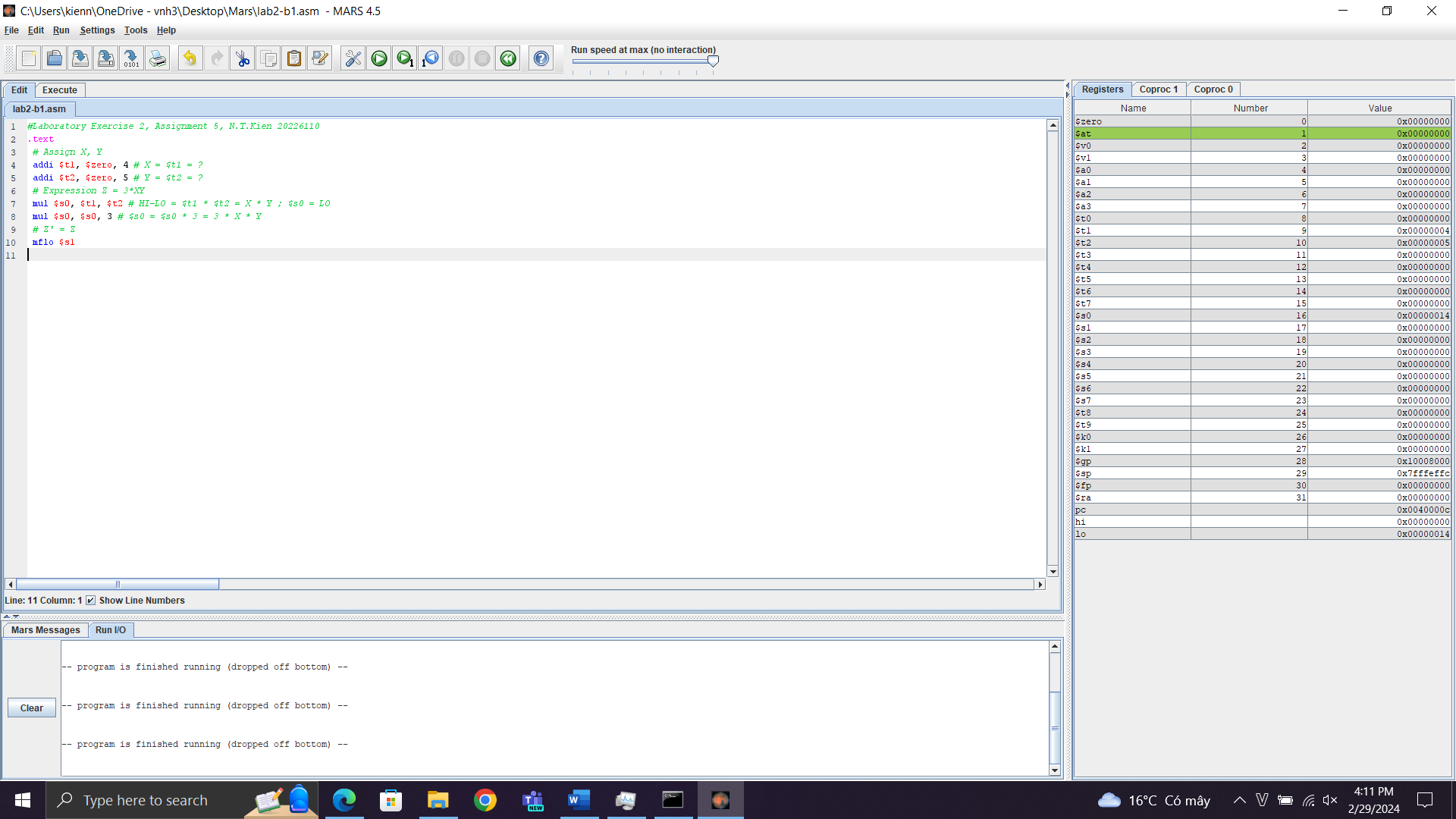
+ Lệnh thứ ba: 0x01298020 => 0000 0001 0010 1001 1000 0000 0010 0000

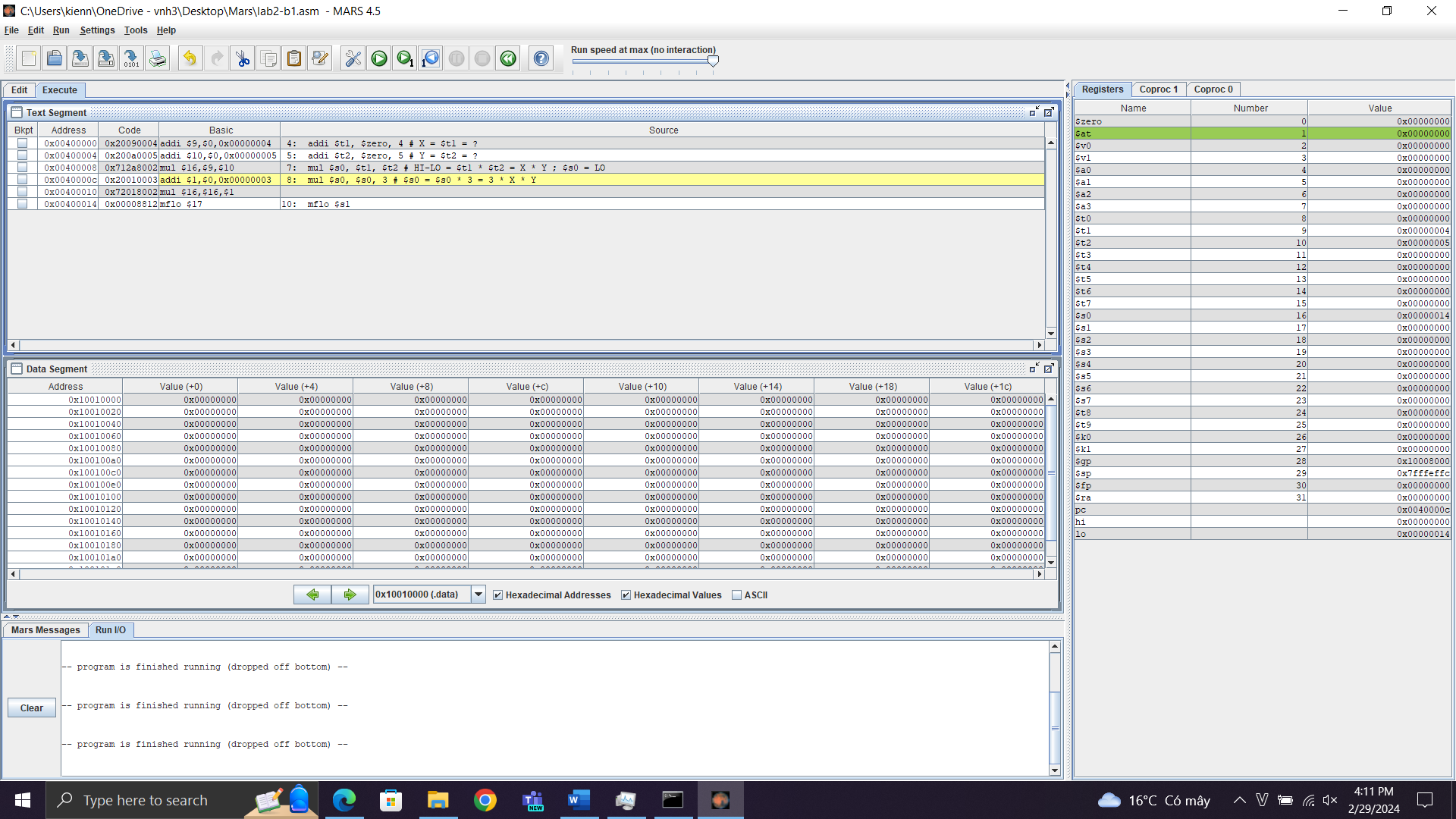
Opcode: 000000 => 0 rs: 01001 => $9 rt: 01001 => $9 rd: 10000 => $16 sh: 00000 fn: 100000 => add $16, $9, $9

+ Lệnh thứ tư: 0x020a8020 => 0000 0010 0000 1010 1000 0000 0010 0000

Opcode: 000000 => 0 rs: 10000 => $16 rt: 01010 => $10 rd: 10000 => $16 sh: 00000 fn: 100000 => add $16, $16, $10

Assignment 5





* Nhận xét:

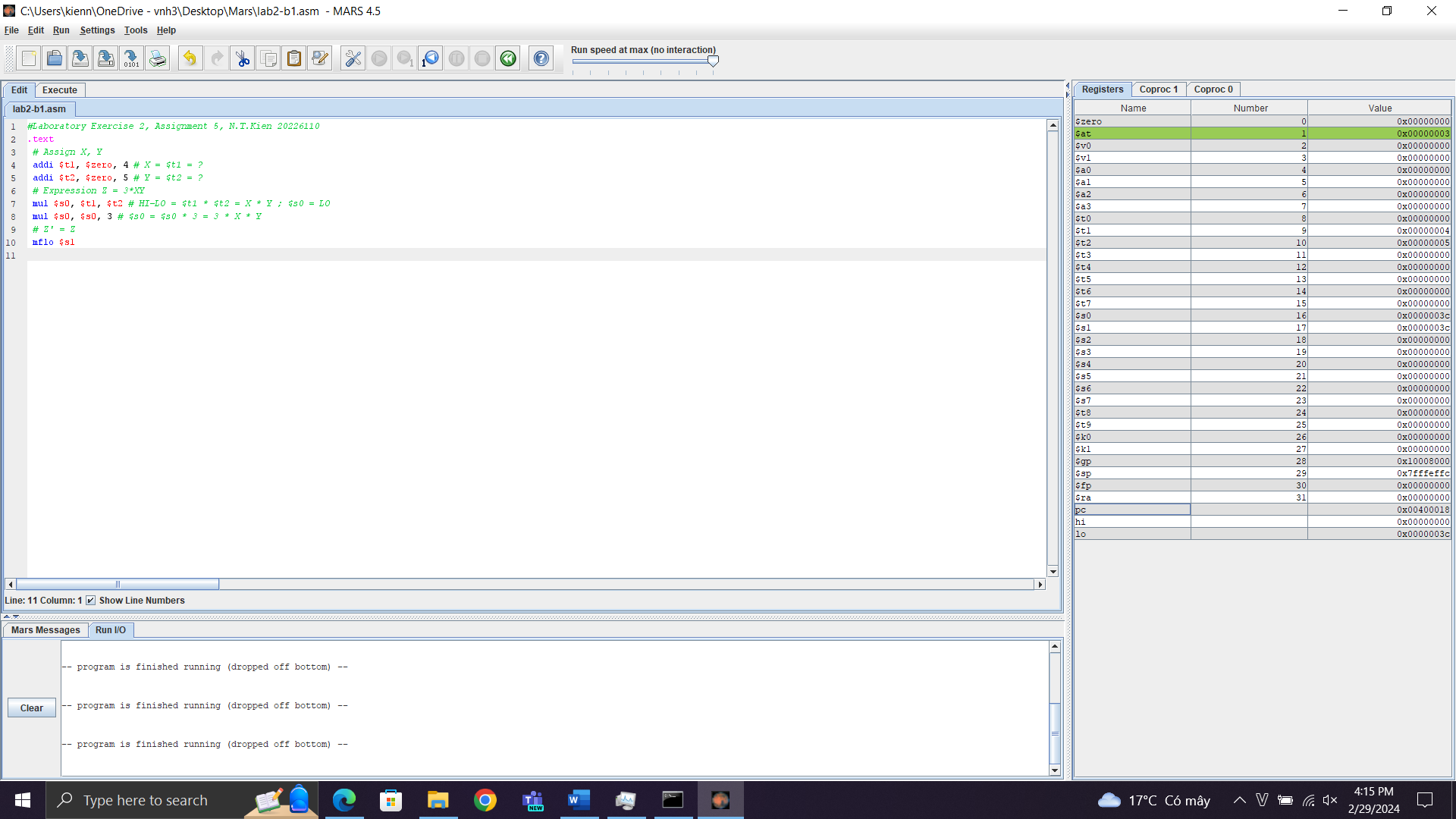
Lệnh mul thứ hai bị tách thành lệnh addi và lệnh mul. Lý do là lệnh mul nhân thanh ghi với một số trực tiếp, nó phải tách thành 2 lệnh:

+ Lệnh addi để lưu số vào 1 thanh ghi (ở đây là $at).

+ Lệnh mul sau khi tách để nhân 2 thanh ghi với nhau. Thanh ghi $t1, $t2 dùng để gán giá trị 4 và 5, thanh ghi $s0 lưu các giá trị cần tính. (lệnh mul đầu là 0x00000014, sau lệnh mul thứ hai là 0x0000003c).

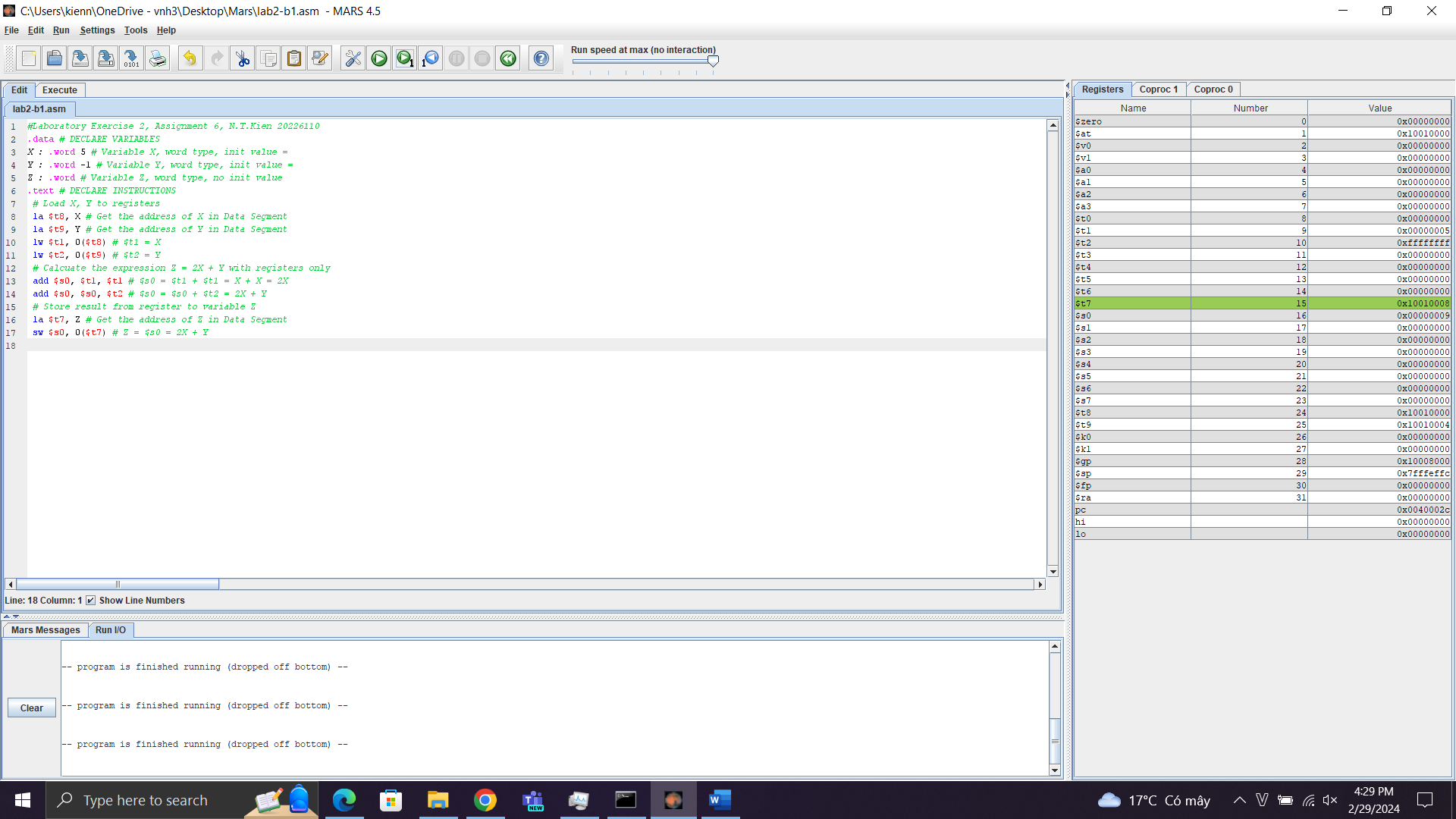
+ Thanh ghi lo thay đổi thành 0x0000003c, thanh ghi hi không thay đổi.

Giá trị các thanh ghi sau khi kết thúc:

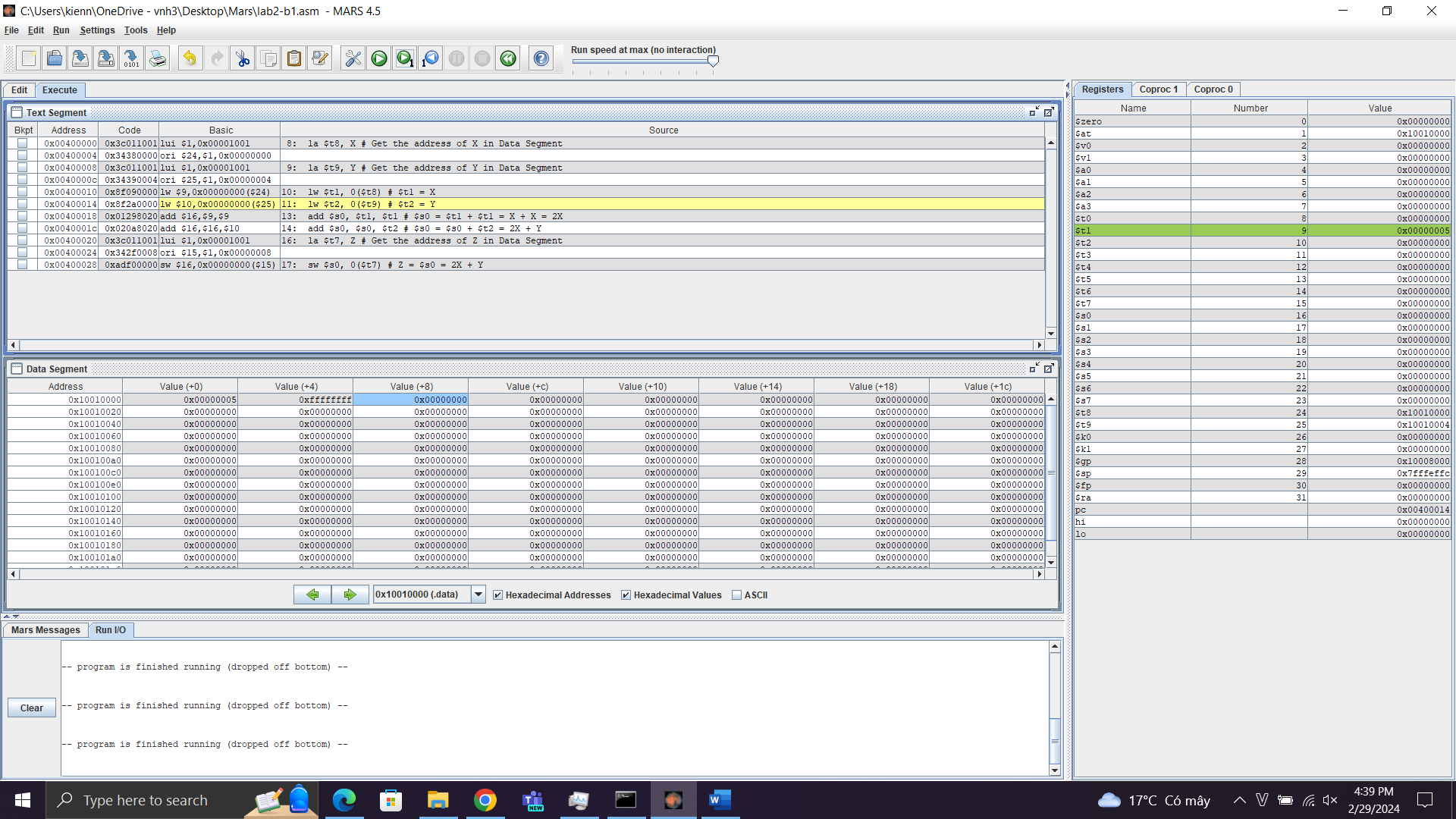


* Đúng với lý thuyết

Assignment 6



* Lệnh la được biên dịch thành 2 lệnh lui và ori để gán 1 số 32 bit vào thanh ghi.
* Khi biên dịch lệnh la thành mã máy, các biến X, Y, Z bằng hằng số.



Sau lệnh thứ nhất: Thanh ghi $at từ 0x00000000 thành 0x10010000

Sau lệnh thứ 2: Thanh ghi $t8 thay đổi từ 0x00000000 thành 0x10010000

Sau lệnh thứ 3: Thanh ghi $t9 thay đổi từ 0x00000000 thành 0x10010004

Sau lệnh thứ 4: Thanh ghi $t1 thay đổi từ 0x00000000 thành 0x00000005

Sau lệnh thứ 5: Thanh ghi $t2 thay đổi từ 0x00000000 thành 0xffffffffffff

Sau lệnh thứ 6 và 7: Thanh ghi $s0 thay đổi từ 0x00000000 thành 0x0000000a sau lệnh thứ sáu, từ 0x0000000a thành 0x00000009 sau lệnh thứ bảy

Sau lệnh thứ 8: Thanh ghi $t7 thay đổi từ 0x00000000 thành 0x10010008

* Thanh ghi pc luôn tăng một khoảng có giá trị 0x00000004 sau mỗi lệnh

Lệnh lw: Để đọc word dữ liệu 32-bit từ bộ nhớ đưa vào thanh ghi.

Lệnh sw: Để ghi word dữ liệu 32-bit từ thanh ghi đưa ra bộ nhớ.

Tìm hiểu thêm các lệnh lb và sb:

Lệnh lb: Đọc 1 byte từ bộ nhớ vào thanh ghi đích.

Lệnh sb: Ghi 1 byte từ thanh ghi vào bộ nhớ.

\_\_\_\_\_HẾT BÀI\_\_\_\_\_